Examining the Annealing Schedules for RNA Design Algorithm

Halid Emre Erhan*, Sinem Sav†, Stas Kalashnikov*, and Herbert H. Tsang‡
*School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada
†Faculty of Engineering & Computer Science, Bilkent University, Ankara, Turkey
‡Applied Research Lab, Trinity Western University, Langley, British Columbia, Canada
Email: herbert.tsang@twu.ca

***Abstract*—RNA structures are important for many biological processes in the cell. One important function of RNA are as catalytic elements. Ribozymes are RNA sequences that fold to form active structures that catalyze important chemical reactions. The folded structure for these RNA are very important; only specific conformations maintain these active structures, so it is very important for RNA to fold in a specific way. The RNA design problem describes the prediction of an RNA sequence that will fold into a given RNA structure. Solving this problem allows researchers to design RNA; they can decide on what folded secondary structure is required to accomplish a task, and the algorithm will give them a primary sequence to assemble. However, there are far too many possible primary sequence combinations to test sequentially to see if they would fold into the structure. Therefore we must employ heuristics algorithms to attempt to solve this problem. This paper introduces *SIMARD*, an evolutionary algorithm that uses an optimization technique called simulated annealing to solve the RNA design problem. We analyzes three different cooling schedules for the annealing process: 1) An adaptive cooling schedule, 2) a geometric cooling schedule, and 3) a geometric cooling schedule with warm up. Our results show that an adaptive annealing schedule may not be more effective at minimizing the Hamming distance between the target structure and our folded sequence's structure when compared with geometric schedules. The results also show that warming up in a geometric cooling schedule may be useful for optimizing *SIMARD*.**

I. Introduction

A. RNA Folding and RNA Design

The RNA folding and inverse RNA folding problems are two issues that have been challenging contemporary bioinformaticians. The folding problem refers to the analysis of RNA primary structure and the method in which it folds to form chemically active secondary structures. This prediction is done through the analysis of RNA nucleotide sequences and the method in which they folds to form structures. Notable researchers in RNA secondary structure prediction research are Hofacker *et al.* in the Vienna package [1] and Zuker *et al* [2].

We can think of RNA design as the reverse procedure of RNA prediction. The goal for RNA folding is to determine RNA primary sequence given its secondary structure. Given enough time, theoretically it is possible to algorithmically determine RNA primary structure from secondary structure. However, with our current computational technology, it is not feasible to check every possible conformation simply due to the vast number of possible combinations of nucleotides. Researcher Schnall-Levin has shown that RNA folding is an NP-hard problem [3]. Researchers in the field are currently using other methods for RNA design in order to avoid a prolonged run time. Notably, the heuristic methods are popular choices.

There are many significant applications in the field of molecular biology for RNA design. The function of RNA is directly associated with the physical structure due to the positioning of active sites. The popular lock-and-key model proposed by Emil Fischer for enzymatic structure and function also describes the relationship between the structure and function of RNA. As a result, RNA design would allow researchers and engineers to synthesize RNA for very specific functions. This would have many applications in fields such as medicine and biotechnology where the ability to manipulate material on a microscopic scale would be invaluable.

B. The RNA Secondary Structure Design Problem

RNA strand has four bases: cytosine (C), guanine (G), adenine (A), and uracil (U). These bases are attached to a sugar-phosphate backbone to form a structure. The strand begin from the 5' to 3' distinct ends. These bases will form hydrogen bonds with each other. The bonding of C with G and A with U are called the Watson-Crick base-pairing. In addition, there is a *wobble* pairs of G and U.

RNA secondary structure is described by the bases that are paired. The secondary structure of a strand of length *n* is a set of pair (i, j), where i and j are in the range of $[1, ..., n]$. The pair (i, j) represents a pairing between the ith and jth bases in the strand. In the secondary structure, each pair can only have one partner.

RNA secondary structure prediction problem can be stated as follows. RNA sequence X when folded into the secondary structure S. Let Φ denote a function that assigns to each RNA sequence X a secondary structure S^* that minimized free energy $E(X, S)$ over all possible secondary structure S of X. Given an RNA sequence X, determine $\Phi(X)$ [4]. .

RNA secondary structure design problem can be stated as follows. Given an RNA secondary structure S^*, find the sequence X^* such that

$$\Phi(X^*) = S^* \tag{1}$$

978-1-5090-0623-6/16/$31.00

During the search for X^*, we determine the quality of a candidate solution X by comparison of the structure $S = \Phi(X)$ with the desired structure S^*. Given the distance metric d, our job is to minimize $d(S, S^*)$ [4].

C. RNA Design - Evolutionary Algorithms

There are many unique approaches available for the implementation of RNA design. Among these approaches is the use of evolutionary algorithms. This is thought to be an effective solution because it avoids traversing the entire set of possible solutions, and instead uses heuristics to solve the problem in a reasonable amount of time.

Evolutionary algorithms mimic the natural process of evolution and natural selection by applying Darwinian principles to find the optimal, or fittest, solution. All of these algorithms apply four important concepts through a number of iterations: reproduction, competition, variation and selection [5]. **Reproduction** refers to individual solutions being passed to the next iteration of the algorithm. **Competition** refers to the process of favouring certain solutions in the progressive iterations of the algorithm based on the fitness level of the solution. **Variation** refers to the processes that introduce variation into the population of individual solutions. These processes are usually mutation or crossover; where mutation is the random replacement of certain elements of the solution with other elements and crossover is the combination of fit solutions. And lastly, **selection** is the process by which the least fit individuals are replaced by newer individuals that result from mutation and crossover.

When it comes to RNA design, there are two major approaches to applying evolutionary algorithms. There is either a focus on recombination with genetic algorithms, or there is a focus on mutation with evolutionary algorithms [5]. Genetic algorithms are more widespread since recombination is a more reliable method of introducing variation than mutation; it is less random. The genetic algorithms in the current literature are: MODENA [6], Frnakestein [7] and GGI-FOLD [8]. An evolutionary algorithm in the current literature is ERD [9].

All of these algorithms apply four fundamental steps: (1) Population initialization; (2) selection; (3) genetic operators (such as mutation and recombination); and (4) termination. The differences between the algorithms are in the implementations of each step.

MODENA is a multi-objective genetic algorithm. This means that it uses multiple factors for determining the fitness of solutions. It aims to minimize the structural distance of the solution from the target structure as well as maximize the free energy of the solution [6]. GGI-FOLD is also multi-objective. It looks at five factors for maximizing fitness: (1) Minimizing the proportion of AU to GC pairs; (2) increasing uniqueness of sequences; (3) avoiding continuous repeats; (4) hybridization; and lastly (5) minimizing the free energy of the structure [8]. Along with this difference in fitness definition, GGI-FOLD also employs the Gibbs sampling method for the selection process [8].

Frnakestein, on the other hand, uses a completely different method of fitness calculation. It uses eight technical schemes that employ Boltzmann probabilities to determine positional fitness. This method maximizes fitness by looking at certain positions or substructures and mutating unfit positions and recombining high fit positions [7].

There is no comprehensive comparison between the three genetic algorithms. However, when Frnakestein was compared with MODENA, Frnakestein suffers from a very long run-time while achieving the same level of accuracy. They used Rfam structures for comparison, which resulted in 23 correct structures out of 29 [7]. This long run-time is due to Frnakestein calculating Boltzmann probabilities, which are computationally costly, as well as the fact that Frnakestein is written in Python [7]. However, Frnakestein designs RNA with nucleotide distributions closer to natural nucleotide distributions while MODENA favours GC pairs since they are more energetically favourable [7]. GGI-FOLD, on the other hand, claims to have a low run-time because it avoids using the costly prediction algorithm provided by the Vienna package, RNAfold, when analyzing results [8].

Ultimately, Tadena asserts that the biggest advantage of genetic algorithms is that they provide more than one solution with varying free energies, allowing the selection of a solution with a lower free energy if it is required [6]. For example, riboswitch structures require lower free energy to allow them to fluctuate between states.

ERD is an evolutionary algorithm the emphasizes mutation over recombination [9] [10]. ERD was designed to take into consideration energy and structural constraints in order to provide structures that are closer to natural structures both in terms of structural distance and free energy [9]. While most algorithms also include the ability to specify structural constraints, the main advantage ERD provides is the ability to include energy constraints; that is, the designer can specify minimal and maximal free energy for the structure.

There is an important initial decomposition process that occurs in ERD before the evolutionary algorithm is applied [9]. The decomposition assigns compatible sequences to substructures for the target structure by looking at substructures from the STRAND database, while still maintaining structural and energy constraints provided in the initial specifications [9]. This structure is intended to act as an initial structure for the evolutionary algorithm [9]. While this initial structure is compatible with the target structure, it does not necessarily fold into the target [9]. The decomposition is done through a modified hierarchical decomposition where multiloops are recognized as distinct elements that make up the substructures [9].

The evolutionary algorithm uses the same four steps that the genetic algorithms use. However, instead of recombination, the algorithm uses mutation to apply variation [9]. ERD initially folds the initial sequence using the prediction function of the Vienna package [1]. The resulting structure is compared with the target; the substructures are compared as well as the thermodynamic free energy. Differing substructures are

replaced using a targeted mutation where the replacing substructure maintains nucleotide length and structure type [9]. It is important to note that mutation only acts on this substructure level and not on the nucleotide level; this maintains consistency between the structures. All combinations of changed substructures are considered as the population. The fittest three structures with the lowest free energy are selected to be used in the next iteration of the algorithm [9]. The algorithm terminates when either 250 iterations pass, or the structural distance between the target structure and the best structure is minimal [9].

ERD is claimed to be faster than MODENA on all accounts: ERD provides more accurate and faster results [9]. This is due to ERD avoiding prediction as much as possible in the initialization step of the algorithm. ERD also claims to outperform MODENA, INFO-RNA, NUPACK and RNAiFOLD when looking at longer sequences of RNA [9]. One large advantage ERD provides is the ability to specify energy constraints, which allows the solution to more accurately mimic natural structures [9].

D. Simulated Annealing

Simulated annealing (SA) is an optimization concept inspired by the annealing of metals, a concept from materials science. It was proposed initially as an algorithm optimization method by Kirkpatrick *et al.* in 1983 [11]. He asserts that there is a "deep and useful" connection between "statistical mechanics", such as the annealing of metals, and "combinatorial optimization". SA was designed to avoid local optima in a given search space by probabilistically accepting less fit options in the beginning of an algorithm. This allows the algorithm to explore the search space. As the algorithm runs, however, the probability of accepting a worse solution decreases, steadily turning the algorithm greedier. This probability is determined by the temperature of the simulated annealing algorithm and the difference in fitness between sequential solutions, and the rate at which the probability decreases is determined by the cooling schedule. Each temperature level represents a slightly different search space, known as a neighbourhood. Solutions found at a certain temperature are said to be neighbours.

SA has been used to solve various computational complex problems. For example, fleet assignment problems for a German airline [12], for school timetabling problems [13] and for the similar RNA folding problem [14] [15] [16]. These are all very computationally taxing problems that require effective heuristic computation to solve.

There are also many different types of cooling schedules. The chosen cooling schedule can play a large role in changing the efficiency of the annealing. Perhaps the simplest cooling schedule is the geometric cooling schedule. After an initial temperature is set, the temperature is reduced by a constant factor at each cooling step. Many researchers have examined various cooling schedules. Abramson *et al.* discuss six different schedules in their paper. In addition to the geometric cooling schedule, they examine a "reheating" concept, where the SA temperature may increase through the annealing process. In addition, they examine an SA process which separates cooling schedules [13]. Similarly, Aarts *et al.* have implemented an adaptive cooling schedule. The cooling schedule sets an initial starting temperature, and the subsequent temperature decrements based on various parameters of the particular problem. It adapts to the mean difference between values, among other parameters [17].

This paper describes and presents a novel algorithm for RNA secondary structure design based on Simulated Annealing (SA). The objectives of this paper are as follows:

- To present a permutation-based simulated annealing algorithm for RNA secondary structure prediction based on the minimization of Hamming distance of the solution from the target structure.
- To compare two simulated annealing cooling schedules: 1) Geometric cooling schedule; and 2) Aarts's adaptive cooling schedule by testing and comparing them on three individual known native structures from the STRAND database.

II. Method

A. SIMARD

SIMARD (Simulated Annealing RNA Design) is a RNA design algorithm. This algorithm finds RNA sequences from a given target structure based on the simulated annealing framework. The motivation behind using SA is that SA has been mathematically proven to eventually reach the global optimum for a given finite combinatorial problem by Geman and Geman [18]. The RNA design problem fits this type of problem perfectly.

SIMARD consists of a number of steps: 1) initialization; 2) mutation; and 3) fitness evaluation.

1) The initialization step is based off of the initialization of ERD [9], an evolutionary algorithm developed by Esmaili-Taheri and Ganjtabesh. The target structure is decomposed into substructures based on multiloops in the target structure. Then subsequences are selected based on the type of corresponding substructure. For example, hairpin subsequences are distributed to hairpin substructures, while stem subsequences are distributed to stem substructures. These substructures are gathered from a database of natural RNA sequences, the STRAND database [19]. This initial sequence generally does not fold into the target structure, but acts as the initial population to be improved for by the evolutionary algorithm.

2) In the mutation step, one subsequence is randomly selected, and replaced by another appropriate subsequence from our database of subsequences. Then the sequence is folded using the fold function from the Vienna package [1].

3) The fitness evaluation step is performed using Hamming distance. The Hamming distance of the resulting structure is compared with the previous solution in the algorithm. The algorithm always keeps structures with lesser Hamming distances for the next iteration. If the Hamming distance is greater than the previous structure, then the structure is kept

if the probability of accepting is high enough. This probability is determined by the expression:

$$Probability[Accept] = e^{\frac{-\Delta Distance}{Temperature}} \quad (2)$$

where $distance$ refers to the change in Hamming distance of the structure from the target structure, so that $-\Delta Distance$ refers to the change in Hamming distance between the new neighbour and the old neighbour. $Temperature$ refers to the temperature of the SA.

The pseudocode for this algorithm is described in Fig. 1. As the algorithm runs, the probability for accepting a worse sequence decreases. In the SA algorithm, the temperature parameter decreases according to a predetermined cooling schedule.

```
1: Sequence = InitialSequence;
2: Temperature = InitialTemperature;
3: Distance = HammingDistance(Sequence, Structure);
4: while (Temperature > FinalTemperature) do
5:    for (i = 1 to NumberOfIterations) do
6:       NewSequence = Mutate(Sequence);
7:       NewDistance = HammingDistance(NewSequence,
         Structure);
8:       Δ Distance = NewDistance - Distance;
9:       if (Δ Distance ≤ 0) OR (with Probability[Accept] =
         e^(-ΔDistance / Temperature)) then
10:         Distance = NewDistance;
11:         Sequence = NewSequence;
12:      end if
13:   end for
14:   decrease Temperature;
15: end while
16: FreeEnergy = FoldAndEvaluate(Sequence);
```

Fig. 1. Structure of the simulated annealing algorithm in RNA secondary structure design

B. Cooling Schedules in SIMARD

One of the most difficult parts of implementing SA is deciding which cooling schedule to use. It is important to have an optimal cooling schedule, as the annealing should aim to move off of local optima, but aim to stay on the global optimum.

In the standard SA implementation, which is a homogeneous Markov chains of finite length. The chains are generated at decreasing temperatures. For any annealing schedule, we have to specify a) an initial temperature T_0; b) a final temperature T_{final} or a stopping criterion; c) the Markov chain length; and d) a set of rules to decrease the temperature.

Beginning with a configuration S, the Metropolis procedure simulates an equilibration process for a fixed temperature T, usually over a large number of time steps. To simulate the cooling procedure, we will repeat the Metropolis procedure for decreasing temperatures, i.e.

$$T_o > T_1 > ... > T_{final} \quad (3)$$

which produces gradually decreasing free energy $\Delta G(S)$ of the configurations

$$\Delta G(S_0) \geq \Delta G(S_1) \geq ... \geq \Delta G(S_{final}) \quad (4)$$

Every combinatorial problem has a particular optimal annealing schedule that fits parameters of the problem. For example, some problems have moves (i.e. single combinatorial permutations) that greatly change the cost function, while other problems may have many more local minima than other problems.

In general, a number of features that signify an effective SA cooling schedule are [20] [21]:

- A high enough initial temperature for a high acceptance probability;
- a low enough termination temperature for a low acceptance probability;
- a low cooling rate, as described by $T_n = \alpha T_{n-1}$ where $\alpha \in (0.8, 1]$ and T_n is the temperature at move n;
- there are at least as many moves as there are neighbouring solutions (i.e. solutions that can be reached within one move.)

The novelty of SIMARD comes from the role simulated annealing plays in the evolutionary process of RNA design. The evolutionary algorithm in SIMARD uses SA to determine whether a new sequence should be kept for the next iteration of the algorithm. When the temperature of the annealing process is high, suboptimal sequences with structures that have large Hamming distances from our target structure may be selected. But as the evolutionary algorithm runs, the probability of selecting bad sequences decreases depending on the annealing.

This paper compares two different cooling schedules for RNA design with *SIMARD*: 1) Geometric cooling schedule and 2) Aarts's adaptive cooling schedule [17].

1) Geometric Schedule: The geometric cooling schedule is a simple schedule based off a constant cooling factor. After an initial temperature is set, the temperature is reduced by a constant factor α, as described by:

$$T_n := \alpha T_{n-1} \text{ with } 0 < \alpha < 1 \quad (5)$$

where T_n is the temperature at time n.

2) Adaptive Schedule: Since the RNA design problem is a complex combinatorial optimization problem, the optimal cooling schedule will not be a simple cooling schedule like the geometric schedule. As a result, researchers have developed cooling schedules that work to adapt to particular problems. An ideal schedule will aim to minimize run time, while still achieving the global minimum. Aarts *et al.* have developed an annealing schedule that aims to do exactly this. Aarts's adaptive cooling scheduler has two parts. A warm up process that tries to find the optimal temperature to start the annealing process and a cooling process that adaptively cools the temperature [17]. This warm up process can also be used to find an optimal starting temperature for the geometric cooling schedule. The warm up process first starts at a temperature

of zero and m_0 iterations are made, where m_0 is the average number of neighbors in of a particular solution. Each iteration updates the temperature according to this equation:

$$T = \overline{\Delta C^+} \left(ln \frac{m_2}{m_2 \chi_0 - (1 - \chi_0)} \right)^{-1} \quad (6)$$

where $\overline{\Delta C^+}$ is the mean value of the difference between the Hamming distances of all worse solutions, χ_0 is the preset acceptance ratio, and m_1 is the number of better neighbours while m_2 is the number of worse neighbours for a solution. The initial temperature for the SA then starts at the final value of T, after m_0 iterations. After the warm up, the temperature cools according to the equation:

$$T_n = T_{n-1} \left(1 + \frac{ln\,(1+\delta)\,T_{n-1}}{3\sigma\,(T_{n-1})} \right)^{-1} \quad (7)$$

where T_n is the temperature of the annealing after n iterations, $\sigma\,(T_{n-1})$ is the standard deviation of the Hamming distances at the current temperature, and δ is the distance parameter that controls the rate of cooling [17].

III. Test Data

We have used four RNA sequences from ERD's RNA data set which were gathered from the RNA STRAND database [9] [19]. Table I summarize the details of our test data-set, the sequences were chosen because they are short sequences that do not take much computational time to design (length range from 300 to 600 nucleotides). This simplifies the experiments so that the run time for each sequence would be reasonable.

TABLE I
Experimental RNA Sequences and Their Respective Lengths

Sequence	Sequence Length (nucleotides)
AF107506	337
AF141485	473
AJ011149	376
AJ130779	506

IV. Results

Table II describes the three experimental conditions for each experiment on these sequences. We have used two types of annealing schedules (geometric and adaptive) and there are two types of modification made (with warmup and without warmup) for the geometric schedulers. This is the warm up process described by Equation 6.

TABLE II
The Experimental Conditions

Annealing Schedule	Modification
Geometric	Without warm up
Geometric	With warm up
Adaptive	With warm up

Fig. 2 shows *SIMARD* runs on these sequences. For each of the figures, one can distinctly see the simulated annealing process; since the probability of acceptance is much higher early in the annealing, there is a much greater variety in the accepted Hamming distance early in the process. However, as the number of moves increases, the variance between accepted Hamming distances increases. This is due to the temperature for the SA decreasing, thus decreasing the probability of accepting worse solutions. Also, we can note the convergence behavior in the annealing algorithm. Clearly the SA run with the adaptive cooling schedule converges later than the other schedules in all of the graphs, while maintaining greater variation between solutions. This is due to the difference in temperature between the schedules. The difference in the schedules is exemplified especially in Fig. 2a and Fig. 2b. Notice how around $15,000$ moves there is a spike in Hamming distance. This is likely due to the adaptive part of Aarts's adaptive cooling schedule adjusting to the rapidly decreasing Hamming distance. As described in Equation 7, the adaptive scheduler adjusts to the standard deviation of Hamming distances at the current temperature before deciding the next temperature.

Table III confirms the previous observations, Aarts's adaptive scheduling required a greater number of moves to converge when compared to both geometric schedules, except for the longest RNA sequence. However, once the annealing has terminated, the final Hamming distance for the geometric scheduling was consistently better than the adaptive scheduler (Table IV). The normal geometric scheduler terminated at a Hamming distance of 0 for every run, while the geometric scheduler with the warm up process consistently came very close to 0.

TABLE III
Comparison of Number of Moves to Convergence for *SIMARD* on Four RNA sequences

	Number of Moves		
Sequence	Adaptive	Geometric	Geometric (with warm up)
AF107506	18,769	15,028	**10,979**
AF141485	20,024	**16,760**	17,719
AJ011149	16,014	15,437	**12,436**
AJ130779	**14,267**	20,397	16,791

TABLE IV
The Hamming distance results of the experiments.

	Best Hamming Distance		
Sequence	Adaptive	Geometric	Geometric (with warm up)
AF107506	**0**	**0**	**0**
AF141485	10	**0**	2
AJ011149	2	**0**	**0**
AJ130779	25	**0**	2

Table II summarizes the three type of annealing schedule setups. Figure 3 shows the three cooling schedules setup we have tested in this study. Since the adaptive cooling schedule does not cool at a constant rate and aims to explore every neighbourhood sufficiently, the increased move number is probably due to its attempt to do this comprehensive exploration. Indeed, the termination temperature for the adaptive

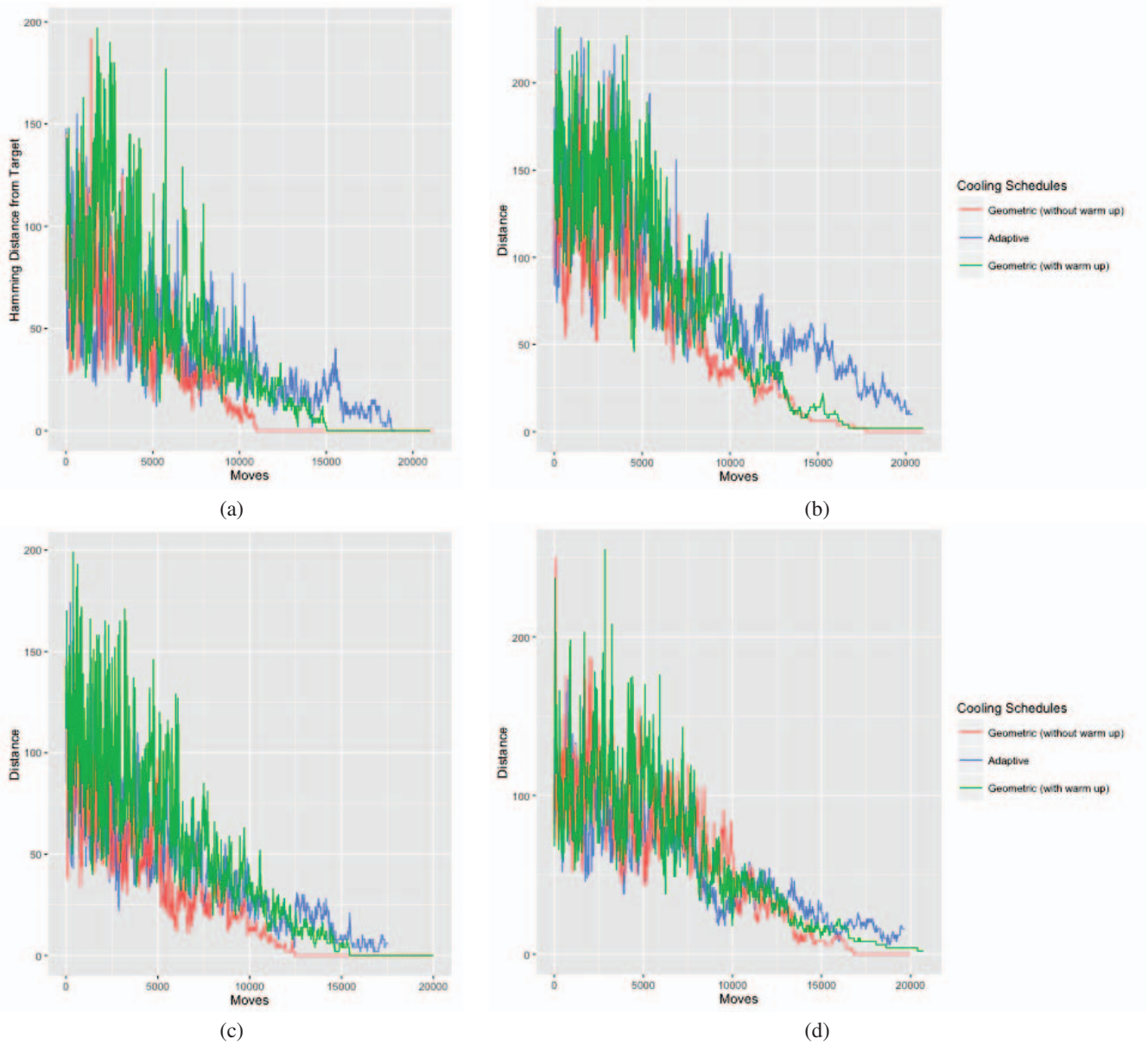

Fig. 2. Plots of comparison of all three annealing schedules on run of the various test RNA sequences. (a) AF107506, a partial sequence of a ribosomal RNA gene from bacterium SY2-21 16S. (b) AF141485, a partial sequence of a 16S ribosomal RNA gene from a eubacterium clone CRE-FL72. (c) AJ011149, a partial sequence of a 16S ribosomal RNA gene from a eubacterium clone 3-25 (d) AJ130779, a partial sequence of a 16S ribosomal RNA gene from a *Stenotrophomonas sp.*

cooling schedule is also consistently higher than the temperature for the geometric schedules (Table V). This indicates that the termination conditions for the adaptive scheduler may be modified to allow the SA to cool to a lower temperature before terminating. This should increase run time, but allow for solutions with greater fitness. In any case, these experiments have shown that the adaptive scheduler requires greater computational resources than the geometric schedulers, which confirms the findings of Tsang *et al* in their experiments on SA in RNA folding [14]. An analysis of the warm up process's effect on the geometric scheduler reveals that it may decrease the number of moves it takes to converge at a reasonable value. Table III shows that the warm up process allowed the geometric scheduler to converge considerably quicker when compared to the geometric scheduler without warm up for all of the sequences except AF141485. This is likely due to starting at a more optimal starting temperature in the warm up process. This indicates that the warm up process used in the adaptive scheduler may be effective at optimizing schedulers, like the geometric scheduler.

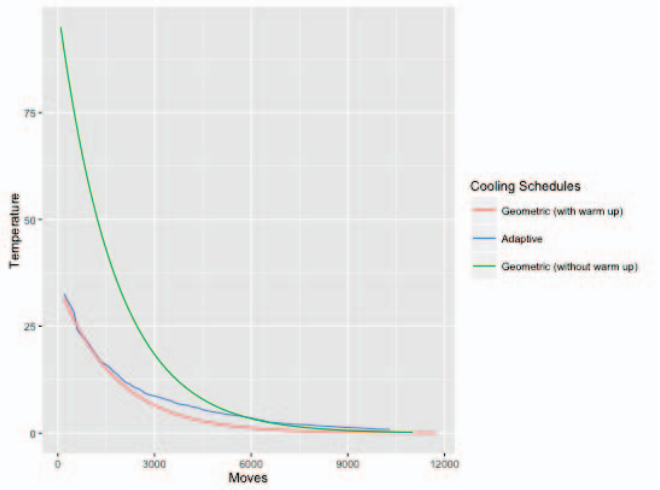

Fig. 3. Comparison of temperatures for all three annealing schedules on AJ011149

TABLE V
COMPARISON OF TERMINATING TEMPERATURES FOR *SIMARD* ON FOUR RNA SEQUENCES

	Terminating Temperature		
Sequence	Adaptive	Geometric	Geometric (with warm up)
AF107506	1.073	0.023	0.121
AF141485	1.041	0.012	0.003
AJ011149	0.912	0.024	0.025
AJ130779	0.841	0.043	0.038

V. CONCLUSION

This study has introduced a new evolutionary algorithm, *SIMARD*, for RNA design. The algorithm employs SA optimization to solve the inverse RNA folding problem through mutating substructures. This paper makes these contributions to the current literature of RNA design: 1) The use of SA in a computational algorithm for RNA design; 2) the use of a substructure based permutation method in SA; and 3) an analysis of different annealing schedules' impact on *SIMARD*.

The analysis of annealing schedules has revealed that the adaptive annealing schedule requires tuning in order for it to be more efficient. However, adaptive annealing ultimately seems to converge in more moves than the geometric schedule due to a non-constant temperature cooling factor. With a goal of Hamming distance minimization, the adaptive scheduler may not be necessary since the geometric schedulers appear to reach a Hamming distance of 0 in less moves anyway. However, a multi-objective evolutionary algorithm, similar to MODENA, that aims to minimize Gibbs free energy in addition to Hamming distance through SA optimization may want to take advantage of an adaptive scheduler [6]. The adaptive scheduler is designed to search more comprehensively through neighbourhoods, so the resulting sequence may have better energy in comparison to a normal multi-objective evolutionary algorithm. In addition, this paper has shown that the warm up process included with Aarts's adaptive annealing schedule is effective at finding an ideal starting temperature. This suggests that other annealing schedules may be improved with a warm up process.

ACKNOWLEDGMENT

The authors would like to acknowledge support from Trinity Western University and Simon Fraser University. In addition, the authors would like to acknowledge the support from Mitacs Globalink program. Also, special thanks to the researchers for their work on ERD which provided part of the software base for *SIMARD*. Additional appreciation to Colin Woodbury and Yoshine Christine Fei for helping with the project.

REFERENCES

[1] I. L. Hofacker, W. Fontana, F. S. Peter, L. S. Bonhoeffer, M. Tacker, and P. Schuster, "Fast Folding and Comparison of RNA Secondary Structures," *Monatshefte fur Chemie*, vol. 125, pp. 167–188, 1994. [Online]. Available: http://fontana.med.harvard.edu/www/Documents/WF/Papers/vienna.rna.pdf

[2] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic Acids Research*, vol. 9, pp. 133–148, 1981.

[3] M. Schnall-Levin, "RNA : algorithms, evolution and design," Ph.D. dissertation, Massachusetts Institute of Technology, 2011. [Online]. Available: http://hdl.handle.net/1721.1/67718

[4] M. Andronescu, A. P. Fejes, F. Hutter, H. H. Hoos, and A. Condon, "A new algorithm for RNA secondary structure design." *Journal of Molecular Biology*, vol. 336, no. 3, pp. 607–624, February 2004.

[5] T. Baeck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. Taylor & Francis, 1997. [Online]. Available: https://books.google.ca/books?id=n5nuiIZvmpAC

[6] A. Taneda, "Multi-objective genetic algorithm for pseudoknotted RNA sequence design," *Frontiers in Genetics*, vol. 3, no. 36, pp. 1–9, 2012.

[7] R. B. Lyngs, J. W. Anderson, E. Sizikova, A. Badugu, T. Hyland, and J. Hein, "Frnakenstein: multiple target inverse RNA folding," *Frontiers in Genetics*, vol. 3, pp. 1–12, 2012. [Online]. Available: http://www.biomedcentral.com/1471-2105/13/260

[8] M. Ganjtabesh, F. Zare-Mirakabad, and A. Nowzari-Dalini, "Inverse RNA folding solution bassed on multi-objective genetic algorithm, and gibbs sampling method," *EXCLI Journal*, vol. 12, pp. 546–555, 2013.

[9] A. Esmaili-Taheri, M. Ganjtabesh, and M. Mohammad-Noori, "Evolutionary Solution for the RNA Design Problem," *Bioinformatics*, vol. 30, no. 9, pp. 1250–1258, 2014.

[10] A. Esmaili-Taheri and M. Ganjtabesh, "ERD: a fast and reliable tool for RNA design including constraints," *BMC Bioinformatics*, vol. 16, no. 1, pp. 1–11, 2015. [Online]. Available: http://dx.doi.org/10.1186/s12859-014-0444-5

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[12] G. Kliewer and S. Tschöke, "A general parallel simulated annealing library and its application in airline industry," in *14th International Parallel and Distributed Processing Symposium (IPDPS 2000)*, 2000, pp. 55–61.

[13] D. Abramson, M. Krishnamoorthy, and H. Dang, "Simulated annealing cooling schedules for the school timetabling problem," *Asia-Pacific Journal of Operational Research*, vol. 16, pp. 1–21, 1999.

[14] H. H. Tsang and K. C. Wiese, "SARNA-Predict: A study of RNA secondary structure prediction using different annealing schedules," in *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2007, pp. 239–246.

[15] ——, "SARNA-Predict: Accuracy improvement of RNA secondary structure prediction using permutation based simulated annealing," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 99, no. 1, pp. 727–740, 2010.

[16] P. Grypma and H. H. Tsang, "SARNA-Predict: Using adaptive annealing schedule and inversion mutation operator for RNA secondary structure prediction," in *IEEE Symposium Series on Computational Intelligence*, 2014, pp. 150–156.

[17] E. H. L. Aarts, F. M. J. de Bont, J. H. A. Habers, and P. J. M. van Laarhoven, "Parallel implementations of the statistical cooling algorithm," *Integration, the VLSI Journal*, vol. 4, no. 3, pp. 209–238, 1986.
[18] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, andthe Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.
[19] M. Andronescu, V. Bereg, H. H. Hoos, and A. Condon, "RNA STRAND: The RNA secondary structure and statistical analysis database," *BMC Bioinformatics*, vol. 9, no. 1, p. 340, 2008.
[20] Y. Li, "Directed annealing search in constraint satisfaction and optimisation," Ph.D. dissertation, University of London, Imperial College of Science, Technology and Medicine, London, 1997.
[21] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: a stochastic approach to combinatorial optimization and neural computing*. Chicester: John Wiley & Sons Ltd., 1989.